Developing Android* Applications with Voice Recognition Features

Android can’t recognize speech, so a typical Android device cannot recognize speech either. Or, is there a way it can?

The easiest way is to ask another application to do the recognition for us. Asking another application to do something in Android is called using intents.

Our target device must have at least one application that can process the Intent for speech recognition, which is called by the RecognizerIntent.ACTION_RECOGNIZE_SPEECH action.

One such app is Google Voice Search. It is one of the best recognizers available for Android and supports a lot of languages. This service requires Internet connection because the voice recognition occurs on Google servers. This app has a very simple Activity that informs users they can speak. The moment the user stops talking, the dialog is closed and our application (intent caller) receives an array of strings with the recognized speech.

A voice recognition sample

Let’s write a little sample app that demonstrates using voice search in applications.

Our application needs to do these things:

- Receive a request for voice recognition
- Check the availability of application for speech recognizing
- If speech recognizing is available, then call the intent for it and receive the results
- If speech recognizing is not available, then show the dialog for installing Google Voice Search and redirect the user to Google Play, if he wants
First, we create a class that implements the logic for speech recognition. Call this class `SpeechRecognitionHelper` where we declare a static, public function `run()` that will receive a request for launching a recognition:

```java
/**
 * A helper class for speech recognition
 */
public class SpeechRecognitionHelper {

/**
 * Running the recognition process. Checks availability of recognition Activity.
 * If Activity is absent, send user to Google Play to install Google Voice Search.
 * If Activity is available, send Intent for running.
 *
 * @param callingActivity = Activity, that initializing recognition process
 */
public static void run(Activity callingActivity) {
    // check if there is recognition Activity
    if (isSpeechRecognitionActivityPresented(callingActivity) == true) {
        // if yes – running recognition
        startRecognition(callingActivity);
    } else {
        // if no, then showing notification to install Voice Search
        Toast.makeText(callingActivity, "In order to activate speech recognition you must install \"Google Voice Search\"", Toast.LENGTH_LONG).show();
        // start installing process
        installGoogleVoiceSearch(callingActivity);
    }
}
}
```

As you can see, besides the `run()` function we need to implement three other functions:
- `isSpeechRecognitionActivityPresented` – checks if the speech recognition application is present on the system
- `startRecognition` – prepares the appropriate Intent and runs the recognition
- `installGoogleVoiceSearch` – initializes the Google Voice Search installation process

To check if the device has an application for speech recognition, we can use the `queryIntentActivities` method in class `PackageManager`. This method gives a list of activities that can process the specified Intent. To receive an instance of the PackageManager, we can use `getPackageManager`.

Our code is shown below:

```java
isSpeechRecognitionActivityPresented
/**
 * Checks availability of speech recognizing Activity
 *
 * @param callerActivity – Activity that called the checking
 * @return true – if Activity there available, false – if Activity is absent
 */
private static boolean isSpeechRecognitionActivityPresented(Activity callerActivity) {
    try {
        // getting an instance of package manager
        PackageManager pm = callerActivity.getPackageManager();
        // a list of activities, which can process speech recognition Intent
        List activities = pm.queryIntentActivities(new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH), 0);

        if (activities.size() != 0) {  // if list not empty
            return true;  // then we can recognize the speech
        }
    } catch (Exception e) {
```
return false; // we have no activities to recognize the speech
}

Now implement the **startRecognition** function. This function will form the appropriate Intent for launching the speech recognition Activity. You can find detailed information for how to do it on [documentation page](#).

Source code:

```java
/**
 * Send an Intent with request on speech
 * @param callerActivity - Activity, that initiated a request
 */
private static void startRecognitionActivity(Activity callerActivity) {
    // creating an Intent with "RecognizerIntent.ACTION_RECOGNIZE_SPEECH" action
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    // giving additional parameters:
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Select an application"); // user hint
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_WEB_SEARCH); // setting recognition model, optimized for short phrases – search queries
    intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 1); // quantity of results we want to receive
    //choosing only 1st - the most relevant
    // start Activity ant waiting the result
    ownerActivity.startActivityForResult(intent, SystemData.VOICE_RECOGNITION_REQUEST_CODE);
}
```

And last, we'll implement the **installGoogleVoiceSearch**. This function will show the dialog, asking the user if he wants to install Google Voice Search and send him to Google Play, if he does.

```java
/**
 * Asking the permission for installing Google Voice Search.
 * If permission granted – sent user to Google Play
 * @param callerActivity – Activity, that initialized installing
 */
private static void installGoogleVoiceSearch(final Activity ownerActivity) {
    // creating a dialog asking user if he want
    // to install the Voice Search
    Dialog dialog = new AlertDialog.Builder(ownerActivity)
        .setMessage("For recognition it's necessary to install \"Google Voice Search\"") // dialog message
        .setTitle("Install Voice Search from Google Play?") // dialog header
        .setPositiveButton("Install", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                try {
                    // creating an Intent for opening applications page in Google Play
                    // Voice Search package name: com.google.android.voicesearch
                    Intent intent = new Intent(Intent.ACTION_VIEW,
                        Uri.parse("market://details?id=com.google.android.voicesearch")); // setting flags to avoid going in application history (Activity call stack)
                    intent.setFlags(Intent.FLAG_ACTIVITY_NO_HISTORY |
                        Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
                    // sending an Intent
                    ownerActivity.startActivity(intent);
                } catch (Exception ex) {
                    // if something going wrong
                }
            }
        });
    dialog.show();
}
```
// doing nothing
}
})

.setNegativeButton("Cancel", null) // cancel button
.create();

dialog.show(); // showing dialog
}

That’s about it. We run the speech recognition Activity. Then request the user’s permission to install Voice Search and send him to Google Play if he consents. One thing we still need to do and that is gather the voice recognition results.

We send a request using the startActivityForResult function to gather results of the launched Activity. We also need to redefine a onActivityResult method in our intent caller Activity. This can be done this way:

// Activity Results handler
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {

// if it’s speech recognition results
// and process finished ok
if (requestCode == SystemData.VOICE_RECOGNITION_REQUEST_CODE && resultCode == RESULT_OK) {

// receiving a result in string array
// there can be some strings because sometimes speech recognizing inaccurate
// more relevant results in the beginning of the list
ArrayList matches = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);

// in “matches” array we holding a results... let’s show the most relevant
if (matches.size() > 0) Toast.makeText(this, matches.get(0), Toast.LENGTH_LONG).show();
}

super.onActivityResult(requestCode, resultCode, data);
}

Now we’re ready

The created class SpeechRecognitionHelper allows us to perform a speech recognition request by calling only one function run().

All that is needed for adding a recognition feature is to add this class in our project and call the run function in needed place. And then implement processing text results by redefining the onActivityResult method for the Activity that initiated the recognition call.

For additional information you can look at the Android Developers website. Here, you’ll find good examples showing how to do voice recognition, and importantly, how to get the available language list. You will need this list if you want to recognize a language other than the user’s default locale.

For fast integration of voice input in to your app, you can download and use this code for the SpeechRecognitionHelper class.

About the Authors
Stanislav works in the Software & Service Group at Intel Corporation. He has 10+ years of experience in software development. His main interest is optimization of performance, power consumption, and parallel programming. In his current role as an Application Engineer providing technical support for Intel-based devices, Stanislav works closely with software developers and SoC architects to help them achieve the best possible performance on Intel platforms. Stanislav holds a Master’s degree in Mathematical Economics from the National Research University Higher School of Economics.
Mikhail is co-author of this blog and an Intel summer intern, who is studying computer science at Lobachevsky University. He likes to deep dive into math and do Android programming tricks.

Notices
INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked “reserved” or “undefined.” Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark® and MobileMark®, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries. Copyright © 2013 Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.